Life Maintenance

AI-Assisted Development Case Study

Author: [Name Redacted for Portfolio Distribution] | Date: 2025 | Platform: Cross-platform mobile (iOS & Android)

Executive Summary

With no formal software development background designed and delivered a fully functional, cross-platform mobile maintenance tracking application using an AI-assisted development workflow. Built over approximately 20 hours across three months, the app addresses a genuine gap in the consumer market — no existing tool comprehensively tracks maintenance across all asset types (home, vehicles, boats, power tools, landscaping equipment) with intelligent task suggestions, cloud sync, and a privacy-first data model. The project demonstrates that disciplined application of AI tools across the full product lifecycle — from concept architecture through backend integration and legal review — can enable non-developers to ship credible software products.

The Challenge

Existing maintenance tracking applications force users to manually create every task from scratch — a time-intensive process that leads to incomplete records and abandoned use. Most apps are scoped to a single asset category (home or vehicle), not the full range of property an owner manages. Additionally, no leading solution offers automatic task suggestions with pre-defined maintenance timelines, multi-asset photo documentation, receipt logging, and genuine data privacy — no data selling, no third-party sharing.

The goal was to build an all-in-one maintenance tracking tool that:

- Spans all asset types — home, vehicles, boats, power tools, landscaping equipment
- Auto-suggests maintenance tasks and timelines by asset type
- Stores service records, contractor details, and receipt photos
- Syncs securely to the cloud while keeping user data private

Approach & Process

The project was executed in deliberate phases, using Claude and Replit as the primary AI-assisted development stack.

Phase 1 — Design & Architecture

Rather than moving immediately to code, significant time was invested in front-end design clarity. A custom Claude prompt was developed to conduct a structured requirements interview, producing a detailed specification covering all screens, features, user flows, and cross-platform requirements (iOS and Android). The architecture was designed with future upgrade phases in mind from the outset — a deliberate decision to avoid costly refactoring later.

Phase 2 — MVP Build

Claude translated the finalized design specification into a build prompt optimized for Replit. The minimum viable prototype was functional on the first attempt. Subsequent iterations followed a structured feedback loop: review all features, document bugs and UI gaps, generate a corrective prompt, rebuild, and repeat until all core functions met original intent.

Phase 3 — Cloud Integration

Firebase was selected for account management and cloud data sync. Claude guided the configuration process step by step — explaining what credentials to extract, how to structure the integration, and how to feed that configuration into Replit. This phase required the most troubleshooting, and the same prompt-iterate-test cycle was applied until sync was stable across devices.

Phase 4 — Polish & Documentation

Claude generated a versioned change log, user agreement, and privacy policy — assets often neglected in early-stage development but essential for credibility and legal compliance.

Key Decisions & Problem Solving

The most consequential judgment call was sequencing cloud sync before the planned OCR receipt-scanning feature. Claude was asked to assess both options across three criteria: alignment with user expectations, implementation complexity, and development time. The analysis supported prioritizing sync — users expect their data to follow them across devices before expecting smart automation.

The most significant business decision was to not publish the app commercially. After a structured risk analysis with Claude covering potential litigation exposure, ongoing platform publishing costs, and realistic revenue projections, the math did not support commercial release without deeper legal infrastructure. This was the right call — and the process of reaching it demonstrates the kind of risk-aware judgment that matters in high-stakes program environments.

In retrospect, the one change worth making at the start would have been a more rigorous market and financial analysis before building — not to change the technical approach, but to enter the commercial viability question with data rather than assumptions.

Outcome

The delivered product is a fully functional, cross-platform mobile application with the following capabilities:

- Multi-asset tracking: home, vehicles, boats, power tools, landscaping equipment
- Automatic maintenance task suggestions and pre-defined service timelines by asset type
- Service record logging with contractor details, cost tracking, and receipt photo storage
- User-uploaded asset photos replacing generic icons
- Cloud-synced user accounts via Firebase with a privacy-first data model
- User agreement and privacy policy

Timeline: ~20 hours of active development spread across 3 months

Demo: YouTube walkthrough (7 minutes)

Lessons & Reflections

The build process itself worked well and would be repeated with minimal changes. The phase-based approach, front-loaded design investment, and structured prompt-iterate-test cycle produced a clean result within time and scope.

The primary lesson is that AI-assisted development demands the same discipline as any other project: the technical build is only one layer of the problem. Market sizing, legal exposure, and commercial viability deserve the same rigorous upfront analysis as the product architecture. Building without that analysis is efficient — it just answers the wrong question first.

AI tools are most powerful when the human operator brings structured thinking to the collaboration. Vague inputs produce vague outputs. The clearer the design intent, the better the build prompts, and the faster the iteration cycles converge on a finished product.

Skills Demonstrated

- End-to-end product development: concept → architecture → MVP → iteration → cloud integration → documentation
- AI tool integration across the full development lifecycle (Claude for design, specification, risk analysis, and documentation; Replit for build and iteration; Firebase for backend infrastructure)
- Phased architecture design with deliberate extensibility for future upgrades
- Structured problem-solving and risk analysis — including a disciplined decision not to commercialize based on legal and financial modeling
- Ability to operate confidently in unfamiliar technical domains through AI-guided learning
- Business and market judgment applied alongside technical execution

---